Práctica 8:

BACKTRACKING

OBJETIVO

- Uso de backtracking en C++.
- Comparación de algoritmos voraces y algoritmos de backtracking.

MATERIAL A ENTREGAR

- *.h, *.cpp
- *.dat
- Makefile
- voraz.png, costes.png, sol.png
- Ejercicios apartado 1. **(previo)**

DESARROLLO

El problema del coloreado de grafos es un clásico problema de computación (http://es.wikipedia.org/wiki/Coloraci%C3%B3n_de_grafos) que consiste en asociar un color a cada uno de los nodos de un grafo de manera que dos nodos vecinos no puedan tener el mismo color y el número de colores empleados sea el mínimo posible. El ejemplo de aplicación más directo es el de coloreado de mapas.

1. Ejercicios previos

Escribe un algoritmo **voraz** (en pseudocódigo o C++) que resuelva el problema para un grafo con los nodos identificados de **0** a **n-1**. La cabecera deberá estar escrita en C++.

Calcula su coste peor en **accesos al grafo**.

2. Resolución mediante algoritmo voraz

El coloreado de grafos se realiza sobre grafos no dirigidos sin pesos. Por ello se deberá modificar el grafo de la práctica pasada para que el peso sea de tipo int (0 no existe arco y 1 si existe) y se deberá crear el método:

void creaGrafoND50();

que genera un grafo no dirigido con un 50% (aproximadamente) del total de arcos posible.

Escribir el programa *voraz.cpp*, que resolverá el problema mencionado en la introducción mediante un algoritmo **voraz**. El algoritmo voraz deberá devolver como resultado un vector con los colores de los nodos y el número total de colores empleados. En la función *main* se deberá crear un grafo de 8 nodos y ejecutar el algoritmo sobre este grafo, para comprobar si funciona correctamente. A continuación calcular el **coste medio** del algoritmo para grafos de 5 a 100 nodos en incrementos de 5. Visualizar el resultado en *voraz.png* **junto con el coste teórico** que calculaste en el apartado 1.

3. Resolución mediante backtracking

Escribir el programa backtracking.cpp, que resolverá el problema mencionado en la introducción mediante un algoritmo de **backtracking con poda**. El algoritmo deberá devolver como resultado un vector con los colores de los nodos y el número total de colores empleado. En la función *main* se

deberá crear un grafo de 8 nodos y ejecutar el algoritmo sobre este grafo, para comprobar si funciona correctamente.

4. Comparación de los algoritmos

En este apartado compararemos los dos algoritmos tanto en coste como en calidad de la solución. Para ello ejecutaremos los dos algoritmos **sobre los mismos grafos**. Se deberá calcular el coste medio en accesos al grafo y el número de colores medio de la solución para los dos algoritmos. El número de nodos del grafo irá desde 5 hasta **un máximo** de 40 (hasta donde permita la velocidad de cálculo del ordenador), en incrementos de 1. Guardar los datos en los ficheros **voraz.dat**, **bt.dat**, **voraz_sol.dat** y **bt_sol.dat**. Representa los datos en las gráficas **costes.png** y **sol.png**.

¿Qué conclusiones obtienes de estas gráficas?

Nota 1:

En esta práctica has utilizado funciones que necesitan precondiciones. Pon todas las precondiciones que sean necesarias mediante el uso de **assert**. También puedes usar **assert** donde estimes oportuno para asegurar la corrección del código.

Nota 2:

A la corrección de la práctica se deberá traer en papel los cálculos realizados para la comprobación de que los algoritmos funcionan correctamente.